Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER

Electronic Notes in Theoretical Computer Science 138 (2005) 3–19

www.elsevier.com/locate/entcs

Decidability of Reachability for Polymorphic Systems with Arrays: A Complete Classification*

Ranko Lazić[1]

Department of Computer Science, University of Warwick, UK

Abstract

Many interesting systems can be seen as having two kinds of state variables: array variables, which are mappings from one data type into another; and basic variables, which are used to control the system, to perform basic computations, and for operations involving arrays.
We investigate such systems where:

- the type of each basic variable is built from type variables using product and sum constructs;
- the type of each array variable is $B \to B'$, where B and B' are types as for basic variables;
- on any type variable, either no operations are available, or only the equality predicate, or only a linear-order predicate;
- type variables denote arbitrary non-empty finite sets.

We present a complete classification of reachability decision problems for these systems into decidable or undecidable.

Keywords: model checking, infinite-state, parameterised, array, decidability

* This research was supported by the EPSRC (GR/S52759/01), and by the Intel Corporation.

[1] Also affiliated to the Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade.

doi:10.1016/j.entcs.2005.08.001

1 Introduction

Systems

Polymorphic systems with arrays (PSAs) were introduced in [10] as a general class of nondeterministic reactive systems. A PSA is polymorphic in the sense that it depends on a signature, which consists of a number of type variables, and a number of symbols whose types can be built from the type variables. Some of the state variables of a PSA can be arrays, which are functions from one type to another.

In the literature on verification of infinite-state systems, there are a number of system classes which are related to PSAs. These include Petri nets and their extensions (e.g. [15]), nested Petri nets [12], broadcast protocols [5] and multi-set rewriting specifications [4]. PSAs are distinguished firstly by having UNITY-style syntax, i.e. with state variables, guards and assignments. Secondly, state variables are typed, and by using more than one type variable, it is straightforward to express systems with more than one parameter. More detailed comparisons can be found in [10].

In this paper, we consider PSAs such that:

(i) the type of each basic variable is built from type variables using product and sum constructs;

(ii) the type of each array variable is $B \to B'$, where B and B' are types as for basic variables;

(iii) on any type variable, either no operations are available, or only the equality predicate, or only a linear-order predicate;[2]

(iv) type variables denote arbitrary non-empty finite sets.

Many interesting systems can be seen as PSAs with the above restrictions. The following are some examples:

- a database with record locking and two security levels of records, where types of record indices and record contents are type variables, and the only operation available on them is equality between record indices [11];
- a fault-tolerant memory, where types of memory addresses and storable data are type variables, and equality is available on both [8];
- any broadcast protocol [5], by using a type variable X to represent the set of all process identifiers, and an array variable $s : X \to \mathit{Enum}_n$ to store the state of each process, where only equality is available on X, and Enum_n is a type consisting of n elements;

[2] An order predicate can express the equality predicate by $t = t' \Leftrightarrow t \leq t' \wedge t' \leq t$.

- a variant of the Bully Algorithm for leadership election in a distributed system [7], by using an array variable $s : X \to \mathit{Enum}_n$, where only a linear-order predicate is available on X [10, Appendix D].

Verification problems

A range of safety properties can be expressed as reachability properties. An instance of reachability consists of a system as above together with specifications of initial and final (i.e. unsafe) states, and the question is whether, for some instantiation of type variables by non-empty finite sets, the system can reach a final state from an initial state. Initial states are specified by requiring that some basic variable has a certain value (i.e. that the system is in a certain control state). To specify final states, we require that the same basic variable has a certain other value.

Type variables on which no operations are available can be used to form types of both basic and array state variables, and of instruction parameters, except that they cannot appear in index types of arrays. To be able to specify that a system manipulates values of such types correctly, we extend reachability as follows. For any type variable X with no operations available, we can add to the specification of final states that they satisfy a formula ϕ_X, which is a Boolean combination of assertions $x \equiv x'$ for state variables $x, x' : X$. Such an assertion $x \equiv x'$ is true if and only if there exists a value v which occurs either in the initial state or in instruction parameters of some transition in the sequence, such that values of x and x' in the final state are copies of v through assignments.

To study decidability of reachability, we divide the class of all PSAs satisfying (i)–(iv) above into subclasses. Each subclass is determined by a *frame*, which specifies a set $\Theta_{\to}$ of typed array variables, and a subset Θ_{rw} of $\Theta_{\to}$. A PSA belongs to the subclass if and only if it has exactly the array variables $\Theta_{\to}$, and on array variables in Θ_{rw} it uses no more than reads and writes (i.e. they are not targets of whole-array operations). For any frame, we have a reachability decision problem, whose instances consist of all PSAs belonging to that frame, together with specifications of initial and final states as above.

Contributions of this paper and comparisons with literature

The results in this paper combine to give a complete classification: we obtain simple (and decidable) conditions such that a frame satisfies them if and only if reachability for that frame is decidable.

The undecidability results are proved by extending [14, Theorem 6.57] and the undecidability theorem in [10]. The latter extension involves a new idea:

requiring that certain arrays may be targets of whole-array operations can be weakened to requiring that certain type variables within index types of those arrays are *controllable*. A type variable X is controllable if and only if there exists an array which may be a target of whole-array operations, whose value type is suitably nontrivial, and whose index type contains a type variable Z such that values of type X can be obtained from values of type Z by a sequence of reads from some arrays.

The main decidability result states that, for any frame such that the types of its arrays form a tree-like structure, reachability is decidable. It is proved by showing that those instances of reachability are instances of the covering problem for well-structured transition systems with effective *pred*-basis and decidable ordering [6].

This result subsumes the decidability results in [14, Chapter 8], [9] and [10]. The decidability theorem in [10] was proved by reduction to a reachability problem for monadic multi-set rewriting specifications with NC constraints, whose decidability was established in [4, Theorem 2]. Now, configurations of such a specification can be represented by arrays of type $X \to \sum_{i=1}^{n} Y$, where the only operations available on X and Y are equality and linear order (respectively), and n is the number of predicate symbols in the specification. [4, Theorem 2] therefore corresponds to this special case of our decidability result. The main difference is that configuration size can vary during a computation, whereas the size of (the set instantiating) X is fixed but arbitrarily large.

In the proof of our decidability result, the greatest part of the constructions of well-quasi-orderings is iteration of product and *Bag* constructs. Such iteration is also done in the proofs of a number of decidability results on nested Petri nets [12]. However, the result in this paper applies to systems which can use linear-order predicates on some type variables, and which can perform whole-array operations. In contrast, tokens in nested Petri nets are unordered, and transfer arcs are not available.

The remaining results in this paper show that, for any frame which does not satisfy the conditions in the undecidability theorem, any instance of reachability can be reduced to an instance of reachability for a frame to which the decidability theorem applies. The reductions are performed by repeated applications of two results.

The first one states that, given any instance of reachability which contains a type variable X on which no operations are available, it can be transformed to an instance without X. The transformation is based on an idea used in [11]: that for data-flow analyses as required in the reachability problems, it suffices to monitor a bounded number of values of type X. However, [11] considered only one array with a simple type, and allowed assigning a value

to all components as the only whole-array operation.

The second result also provides a transformation for removing a type variable X from an instance of reachability. The assumptions are that X does not appear in value types of arrays, and that it does not appear in the index type of any array which may be a target of whole-array operations. Semantically, this transformation performs an abstraction which preserves the answer to the reachability problem. It is based on results in [8] and [14, Chapters 5 and 6], but it extends them by allowing a linear-order predicate on X.

Organisation

In Section 2, we introduce the class of systems we are considering, and define the notion of frame. For any frame, its reachability decision problem is defined in Section 3. Sections 4–6 contain the main results of the paper. In Section 7, we briefly conclude and discuss future work.

The paper is in form of an extended abstract, so proofs are mostly outlines.

2 Systems and frames

Types and terms

For one-step computations, we use a typed λ-calculus. Details of its syntax and semantics can be found in [10] — the following is a summary.

Basic types are built from type variables, products and non-empty sums. There are also function types, from one basic type to another. Function types are used as types of operations such as equality and linear-order predicates, and as types of array state variables.

$$B ::= X \mid B_1 \times \cdots \times B_n \mid B_1 + \cdots + B_{n \geq 1}$$
$$T ::= B \mid B \to B'$$

Terms are built from term variables, tuple formation, tuple projection, sum injection, sum case, λ-abstraction, and function application.

We consider only well-typed terms. A signature consists of a finite set Ω of type variables, and a type context Γ which is a sequence $\langle x_1 : T_1, \ldots, x_n : T_n \rangle$ of typed and mutually distinct term variables, where the types T_i can contain only type variables from Ω. A well-typed term-in-context is written $\Omega, \Gamma \vdash t : T$, where these valid type judgements are deduced by standard typing rules [13].

Using the types and terms above, we can for example express:

- the singleton type *Unit* as the empty product, and its unique element as the empty tuple;

- the boolean type *Bool* as the sum of two *Unit* types, and terms *false*, *true*, and *if* t *then* t'_1 *else* t'_2;
- for any positive n, the n-element enumerated type $Enum_n$ as the sum of n *Unit* types, its elements $e_1, \ldots, e_n$, and a case term.

We can also express any given operation on the *Bool* and $Enum_n$ types, of any arity.

For semantics of types, a finite set Ω of type variables is instantiated by a mapping ω to non-empty sets. For any type T such that $Vars(T) \subseteq \Omega$, its semantics with respect to ω is a non-empty set $[\![T]\!]_\omega$, which is defined in the usual way.

For semantics of terms, a signature (Ω, Γ) is instantiated by an ω as above, and a mapping $\gamma \in [\![\Gamma]\!]_\omega$, i.e. $Dom(\gamma) = Dom(\Gamma)$ and $\gamma[\![x]\!] \in [\![T]\!]_\omega$ for all $x : T$ in Γ. For any well-typed term-in-context $\Omega, \Gamma \vdash t : T$, its semantics with respect to (ω, γ) is an element $[\![t]\!]_{\omega,\gamma}$ of $[\![T]\!]_\omega$, and is defined in the standard way.

Polymorphic systems with arrays

A PSA is a 5-tuple $(\Omega, \Gamma, \Theta, R, I)$, where (Ω, Γ) is a signature, Θ is a type context specifying the state variables and their types, R is a finite set of instructions, and I is a set of instantiations of (Ω, Γ).

The semantics of a PSA $(\Omega, \Gamma, \Theta, R, I)$ is a transition system $(S, \rightarrow)$, where the set of states S consists of all (ω, γ, θ) such that $(\omega, \gamma) \in I$ and $\theta \in [\![\Theta]\!]_\omega$.

For details of the syntax and semantics of PSAs, we refer the reader to [10]. The same paper also contains a variant of the Bully Algorithm for leadership election in a distributed system [7] expressed as a PSA.

As shown in [10, Appendix B], instruction parameters and assignments to array variables can be used to express a range of array operations, including the following:

Multiple partial assign. Assigning $t_1, \ldots, t_n$ to all components x of an array a which satisfy conditions $d_1, \ldots, d_n$ respectively, where x may occur free in the t_i and d_i:

$$a := \lambda x : B \cdot \mathit{if}\ d_1\ \mathit{then}\ t_1\ \mathit{elseif}\ \cdots\ d_n\ \mathit{then}\ t_n\ \mathit{else}\ a[x]$$

We may abbreviate this as $a[x : d_1; \cdots; d_n] := t_1; \cdots; t_n$. Note that if d_i and d_j with $i < j$ overlap, assigning t_i takes precedence.

Write. Assigning $t'_1, \ldots, t'_n$ to $a[t_1], \ldots, a[t_n]$:

$$a[x : x = t_1; \cdots; x = t_n] := t'_1; \cdots; t'_n$$

where x is fresh. We may abbreviate this as

$$a[t_1; \cdots; t_n] := t'_1; \cdots; t'_n$$

Frames

A *frame* is a 4-tuple $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$, where:

- Ω, Γ is a signature;
- $\Theta_{\rightarrow}$ is a type context which is disjoint from Γ, such that $(\Omega, \Gamma\Theta_{\rightarrow})$ is a signature, and which contains only function types;
- Θ_{rw} is a subsequence of $\Theta_{\rightarrow}$;
- I is a set of instantiations of Ω, Γ.

A frame defines a class of PSAs by fixing a signature, fixing array state variables and their types, specifying which of those cannot be targets of whole-array operations, and fixing a set of signature instantiations.

More precisely, given a frame $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$, its class consists of all PSAs $(\Omega, \Gamma, \Theta_{\rightarrow}\Theta_{\not\rightarrow}, R, I)$ such that:

- $\Theta_{\not\rightarrow}$ is a type context which is disjoint from $\Gamma\Theta_{\rightarrow}$, and which contains only basic types;
- for any $a \in Dom(\Theta_{\mathrm{rw}})$, any assignment to a in R is a write.

Systems and frames in this paper

In the rest of the paper, we shall be considering the following systems and frames.

Any signature will consist of three kinds of type variables: with no operations available, with only equality, and with only a linear-order predicate. More formally, signatures will be of the form

$$(\Omega_{\emptyset} \uplus \Omega_{=} \uplus \Omega_{\leq},$$
$$\langle =_X \colon X \times X \to \mathit{Bool} \mid X \in \Omega_{=} \rangle$$
$$\langle \leq_X \colon X \times X \to \mathit{Bool} \mid X \in \Omega_{\leq} \rangle)$$

Without restricting expressiveness, we shall assume that any basic type is a sum of products of type variables, i.e. of the form $\sum_{i=1}^{n \geq 1} \prod_{j=1}^{m_i} X_{i,j}$.

Since an array whose index type is a sum of n types is isomorphic to n arrays, we can also assume without loss of generality that the index type of any array is a product of type variables, i.e. of the form $\prod_{j=1}^{m} X_j$.

Moreover, it makes computational sense to consider in index types only type variables on which at least equality is available, so we shall assume that

each X_j is in $\Omega_= \uplus \Omega_\leq$.

For signatures as above, we are interested in instantiations of type variables by arbitrary non-empty finite sets, and of $=_X$ and $\leq_X$ symbols by equality and linear-order predicates. For type variables in $\Omega_\emptyset$, since they do not occur in index types of arrays, we can simplify and instantiate them by countably infinite sets. Finally, it suffices to consider only one instantiation from each isomorphism class, so any set of instantiations I will consist of all (ω, γ) such that:

- for each $X \in \Omega_\emptyset$, $\omega[\![X]\!] = \mathcal{Z}^+$;
- for each $X \in \Omega_=$, $\omega[\![X]\!] = \{1, \ldots, k\}$ for some $k \in \mathcal{Z}^+$, and $\gamma[\![=_X]\!]$ is the equality predicate on $\{1, \ldots, k\}$;
- for each $X \in \Omega_\leq$, $\omega[\![X]\!] = \{1, \ldots, k\}$ for some $k \in \mathcal{Z}^+$, and $\gamma[\![\leq_X]\!]$ is the usual order predicate on $\{1, \ldots, k\}$.

3 Reachability decision problems

Suppose $(\Omega, \Gamma, \Theta, R, I)$ is a PSA. As assumed in Section 2, Ω is of the form $\Omega_\emptyset \uplus \Omega_= \uplus \Omega_\leq$. As part of instances of reachability which involve this PSA, we want to be able to assert that, for each type variable $X \in \Omega_\emptyset$, a Boolean formula built from assertions of the form $x \equiv x'$ is true in the final state, where $x : X, x' : X \in \Theta$. The interpretation of $x \equiv x'$ is that there exists a value v which occurs either in the initial state or in instruction parameters of some transition in the sequence, such that values of x and x' in the final state are copies of v through assignments.

Since no operations are available on type variables $X \in \Omega_\emptyset$, the above is equivalent to replacing the $\equiv$ relation by equality, and considering only sequences of transitions which are $\Omega_\emptyset$-*fresh*. A sequence of transitions has the latter property if and only if, for each $X \in \Omega_\emptyset$:

(i) values of type X in the initial state are mutually distinct, and

(ii) in any transition, values of type X in the instruction parameters are mutually distinct, and disjoint from values of type X in the source state.

Suppose $(\Omega, \Gamma, \Theta_\rightarrow, \Theta_{\mathrm{rw}}, I)$ is a frame. The $(\Omega, \Gamma, \Theta_\rightarrow, \Theta_{\mathrm{rw}}, I)$-*reachability* decision problem is the following:

Instance: This consists of:

- type context $\Theta_{\not\rightarrow}$ and instructions R such that $(\Omega, \Gamma, \Theta_\rightarrow \Theta_{\not\rightarrow}, R, I)$ is a PSA belonging to the frame;
- a variable $b : \mathit{Enum}_n \in \Theta_{\not\rightarrow}$ and $i, j \in \{1, \ldots, n\}$;

- for each $X \in \Omega_\emptyset$, a formula

$$\phi_X \ ::= \ x = x' \mid \mathit{true} \mid \neg\phi_X \mid \phi_X \wedge \phi'_X$$

where $x : X, x' : X \in \Theta_{\not\to}$.

Question: Does $(\Omega, \Gamma, \Theta_\to\Theta_{\not\to}, R, I)$ have an $\Omega_\emptyset$-fresh sequence of transitions from a state satisfying $b = e_i$ to a state satisfying $b = e_j \ \wedge \ \bigwedge_{X\in\Omega_\emptyset} \phi_X$?

In [10, Appendix D.2], there are several examples of safety properties for the Bully Algorithm PSA expressed as instances of reachability. This PSA belongs to a frame of the form

$$(\{X\}, \langle \leq_X : X \times X \to \mathit{Bool}\rangle, \langle a : X \to \mathit{Enum}_k\rangle, \langle\rangle, I)$$

To express safety properties which require that all components of a equal a certain value in the initial state, the system can be extended to reset a as needed in the first transition.

We now define a simple quasi ordering between frames. $(\Omega, \Gamma, \Theta_\to, \Theta_{\mathrm{rw}}, I)$ is smaller than $(\Omega', \Gamma', \Theta'_\to, \Theta'_{\mathrm{rw}}, I')$ if and only if the latter contains more type variables, it allows linear-order predicates on some type variables on which only equalities were available, it contains more array variables, and it does not require only reads and writes on some array variables which were free of that requirement. Moreover, type variables and array variables can be renamed.

More formally, $(\Omega, \Gamma, \Theta_\to, \Theta_{\mathrm{rw}}, I) \preceq (\Omega', \Gamma', \Theta'_\to, \Theta'_{\mathrm{rw}}, I')$ if and only if there exists an injection ι on $\Omega \uplus \mathit{Dom}(\Theta_{\mathrm{rw}})$ such that:

(i) $\iota(\Omega) \subseteq \Omega'$;

(ii) $\iota(\Omega_\emptyset) = \Omega'_\emptyset \cap \iota(\Omega)$ and $\iota(\Omega_=) \supseteq \Omega'_= \cap \iota(\Omega)$;

(iii) $\iota(\Theta_\to) \subseteq \Theta'_\to$;

(iv) $\iota(\Theta_{\mathrm{rw}}) \supseteq \Theta'_{\mathrm{rw}} \cap \iota(\Theta_\to)$.

Proposition 3.1 *If $(\Omega, \Gamma, \Theta_\to, \Theta_{\mathrm{rw}}, I) \preceq (\Omega', \Gamma', \Theta'_\to, \Theta'_{\mathrm{rw}}, I')$, then $(\Omega, \Gamma, \Theta_\to, \Theta_{\mathrm{rw}}, I)$-reachability is reducible to $(\Omega', \Gamma', \Theta'_\to, \Theta'_{\mathrm{rw}}, I')$-reachability.*

Proof. Let ι be as in the definition of $\preceq$. Suppose $(\Omega, \Gamma, \Theta_\to\Theta_{\not\to}, R, I)$ is a PSA in the class of $(\Omega, \Gamma, \Theta_\to, \Theta_{\mathrm{rw}}, I)$. Let ι' be an injective extension of ι to $\mathit{Dom}(\Theta_{\not\to})$ such that $\iota'(\mathit{Dom}(\Theta_{\not\to}))$ is disjoint from $\mathit{Dom}(\Theta'_\to)$. It is routine to check that $(\Omega', \Gamma', \Theta'_\to\iota'(\Theta_{\not\to}), \iota'(R), I')$ is in the class of $(\Omega', \Gamma', \Theta'_\to, \Theta'_{\mathrm{rw}}, I')$, and that it simulates $(\Omega, \Gamma, \Theta_\to\Theta_{\not\to}, R, I)$. (For any $X \in \iota(\Omega_=)\backslash\Omega'_=$, occurences of $=_{\iota^{-1}(X)}$ in R are replaced by $\leq_X \wedge \geq_X$ in $\iota'(R)$.) □

4 Undecidability results

For types B and B' which are non-empty sums of products of type variables, we write $B \trianglelefteq B'$ if and only if the syntax tree of B is a subtree of that of B'. More formally:

$$\sum_{i=1}^{n} \prod_{j=1}^{m_i} X_{i,j} \trianglelefteq \sum_{i'=1}^{n'} \prod_{j'=1}^{m'_{i'}} X'_{i',j'} \;\Leftrightarrow$$
$$\exists f : \{1,\ldots,n\} \xrightarrow{1-1} \{1,\ldots,n'\} \cdot \forall i \cdot$$
$$\exists g_i : \{1,\ldots,m_i\} \xrightarrow{1-1} \{1,\ldots,m'_{f(i)}\} \cdot \forall j \cdot X_{i,j} = X'_{f(i),g_i(j)}$$

Suppose X_i for $i = 1,\ldots,n+1$ are type variables, and $a_i : B_i \to B'_i$ for $i = 1,\ldots,n$ are typed array variables. We say that $X_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} X_{n+1}$ is a *path* if and only if for each $i \leq n$, $X_i \trianglelefteq B_i \;\wedge\; X_{i+1} \trianglelefteq B'_i$.

A path as above is a *cycle* if and only if $n \geq 1$ and $X_1 = X_{n+1}$.

Given a frame $(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$, we say that a type variable $X \in \Omega_{=} \cup \Omega_{\leq}$ is *controllable* if and only if:

(i) $\Theta_{\to}$ contains a path from some Z to X and,

(ii) for some $c : C \to C' \in \Theta_{\to} \setminus \Theta_{\mathrm{rw}}$,
$$Z \trianglelefteq C \;\wedge\; (\mathit{Bool} \trianglelefteq C' \;\vee\; \exists Z' \in \Omega_{=} \cup \Omega_{\leq} \cdot Z' \trianglelefteq C')$$

Theorem 4.1 *$(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$-reachability is undecidable if any of the following conditions is satisfied:*

(a) $\Theta_{\to}$ contains a cycle;

(b) for some $a : B \to B' \in \Theta_{\to}$ such that $\mathit{Bool} \trianglelefteq B'$, there exist controllable X and X' such that $X \times X' \trianglelefteq B$;

(c.1) for some $a : B \to B' \in \Theta_{\to}$ such that
$$\exists Y, Y' \in \Omega_{=} \cup \Omega_{\leq} \cdot Y \times Y' \trianglelefteq B'$$
there exists controllable X such that $X \trianglelefteq B$;

(c.2) for some $a : B \to B' \in \Theta_{\to}$ such that
$$\exists Y \in \Omega_{=} \cup \Omega_{\leq} \cdot Y \trianglelefteq B'$$
there exists controllable X and some X' such that $X \times X' \trianglelefteq B$;

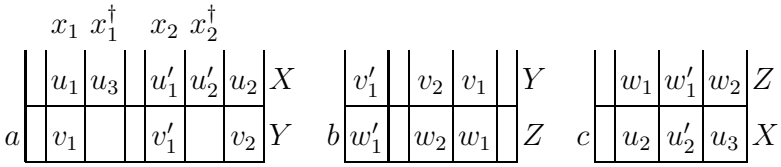
(c.3) for some distinct $a_1 : B_1 \to B'_1, a_2 : B_2 \to B'_2 \in \Theta_{\to}$ such that
$$\exists Y_1, Y_2 \in \Omega_{=} \cup \Omega_{\leq} \cdot Y_1 \trianglelefteq B'_1 \wedge Y_2 \trianglelefteq B'_2$$
there exists controllable X such that $X \trianglelefteq B_1$ and $X \trianglelefteq B_2$;

(d) for some $a : B \to B' \in \Theta_{\to}$ such that
$$\exists Y \in \Omega_{=} \cup \Omega_{\leq} \cdot Y \trianglelefteq B'$$
there exists controllable $X \in \Omega_{\leq}$ such that $X \trianglelefteq B$.

Proof. In each case, we show how to reduce from location reachability for 2-counter machines (as defined in e.g. [10, Appendix C.1]) to $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$-reachability.

If (a) is satisfied, we adapt the proof of [14, Theorem 6.57]. Let $X_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} X_{n+1}$ be a cycle in $\Theta_{\rightarrow}$, so that $n \geq 1$ and $X_1 = X_{n+1}$. To represent values of counters, we use the fact that, starting from a value of type X_1, we can compute another one by reading from the arrays $a_1, \ldots, a_n$ in sequence. For each counter c_i, we use state variables x_i and $x_i^\dagger$. That c_i has value $k_i \in \mathcal{N}$ is represented by needing to perform k_i iterations of reads as above to obtain the value of $x_i^\dagger$ starting from the value of x_i.

The following illustrates representing a configuration of a 2-counter machine in which c_1 and c_2 have values 2 and 1 (respectively), where the cycle consists of arrays $a : X \to Y$, $b : Y \to Z$ and $c : Z \to X$:

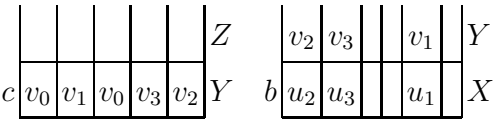


In the remaining cases, we extend the proof of the undecidability theorem in [10]. For (b), the relevant class is $X \times X'$-to-Bool (where X and X' may be the same); for (c.1), it is X-to-Y,Y'; (c.2) and (c.3) are similar; for (d), it is X,$\leq$-to-Y.

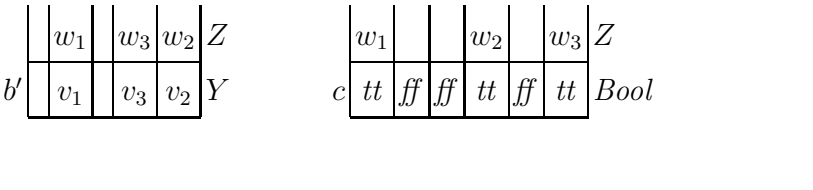
The most complex part of the extension is due to the fact that conditions (b)–(d) are weaker by using the concept of controllability than definitions of the corresponding classes in [10]. In the proof in [10], arrays which are used to represent values of the counters of 2-counter machines are initialised by being reset to certain values at the start of computations. It turns out that such initialisations are not necessary, provided certain type variables which occur in index types of such arrays are controllable. For such a type variable X, we can use other arrays (from the definition of controllability) to represent the set of all values of type X which are currently used in representing the counter values. One of those arrays needs to be outside Θ_{rw} (i.e. able to be a target of whole-array operations), so that the set of values of type X can be initialised to empty.

Suppose $X \in \Omega_{=} \cup \Omega_{\leq}$ is controllable, and let Z, a path from Z to X, and c be as in the definition of controllability. If c is the first array in the path, sets of values of type X can be represented as in the following illustration. Here, the path consists of $c : Z \to Y$ and $b : Y \to X$. c is initialised to contain a value v_0 at all indices. To represent a set of values of type X, the same number of values of type Y is used, which are mutually distinct and distinct

from v_0.



If the path does not begin with c, we can use c to keep a set of values of type Z, and use the path to obtain values of type X from those. The following is an illustration, where the path consists of $b' : Z \to Y$ and $b : Y \to X$, and we have $c : Z \to \mathit{Bool}$.



□

5 Reduction results

The first result in this section states that, for any frame which contains a type variable X on which no operations are available, any instance of reachability can be transformed to an instance for a frame which is obtained by removing X.

Theorem 5.1 *Suppose* $(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$ *is a frame, and* $X \in \Omega_{\emptyset}$. *Then* $(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$*-reachability is reducible to* $(\Omega \setminus \{X\}, \Gamma, \Theta_{\to}[\mathit{Unit}/X], \Theta_{\mathrm{rw}}[\mathit{Unit}/X], I \setminus \{X\})$*-reachability.*

Proof. Consider an instance $\Theta_{\not\to}$, R, $b : \mathit{Enum}_n$, i, j, $\langle \phi_Y \mid Y \in \Omega_{\emptyset} \rangle$ of $(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$-reachability.

Let m be the number of distinct variables in ϕ_X. The first part of the transformation is based on the fact that, at any state in an $\Omega_{\emptyset}$-fresh computation, the values of the variables in ϕ_X originated from some at most m values in the initial state and instruction parameters of previous transitions.

More precisely, the original instance can be transformed to an instance of reachability for the frame

$$(\Omega \setminus \{X\}, \Gamma, \Theta_{\to}[\mathit{Enum}_{m+1}/X], \Theta_{\mathrm{rw}}[\mathit{Enum}_{m+1}/X], I \setminus \{X\})$$

$\Omega_{\emptyset} \setminus \{X\}$-fresh computations from states satisfying the initial condition in the transformed instance correspond to $\Omega_{\emptyset}$-fresh computations from states

satisfying $b = e_i$ in the original instance, transformed as follows. We write 1, ..., m, $\top$ for the elements of $Enum_{m+1}$.

(i) In the initial state, for some $k \in \{0, \ldots, m\}$, some k values of type X are replaced by 1, ..., k, and the rest are replaced by $\top$.

(ii) In any transition, if $k \in \{0, \ldots, m\}$ is the smallest such that values of type X in the source state are replaced by values in $\{1, \ldots, k, \top\}$, then:
  (a) for some $k' \in \{k, \ldots, m\}$, some $k' - k$ values of type X in the instruction parameters are replaced by $k + 1$, ..., k';
  (b) for any value of type X in the source state or the instruction parameters which is replaced by a non-$\top$ value, at least one assignment of it to a value in the target state is performed exactly, and others are transformed to assign $\top$;[3]
  (c) the number of values of type X in the target state which are replaced by non-$\top$ values is at most m.

The formula ϕ_X is incorporated into the transformed system. For any atomic formula $x = x'$, if any of x and x' has value $\top$, then $x = x'$ evaluates to a Boolean value $\top$. The Boolean operators are extended to $\top$ in the usual way.

By (i) and (ii)(c), the number of non-$\top$ values in states of the resulting instance is at most m. Therefore, instead of replacing X in value types of arrays by $Enum_{m+1}$, we can replace it by *Unit* (i.e. remove it) and keep indices of occurences of non-$\top$ values in basic state variables. This gives us the second part of the transformation, where we obtain an instance of reachability for the frame $(\Omega \setminus \{X\}, \Gamma, \Theta_{\rightarrow}[Unit/X], \Theta_{\mathrm{rw}}[Unit/X], I \setminus \{X\})$, as required. □

The following result states that, for any frame and any type variable X on which at least equality is available, if X does not occur in value types of arrays and it does not occur in index types of arrays which can be targets of whole-array operations, then any instance of reachability can be transformed to an instance for a frame obtained by replacing X by an $Enum_n$ type. Since we are assuming that index types of arrays are products of type variables, the latter substitution is an abbreviation: for each occurence of X in an index type, the array is replaced by n arrays in whose index types that occurence of X is replaced by *Unit*.

If $\Theta_{\not\rightarrow}$ is a type context which contains only basic types, and X is a type variable, we write $|\Theta_{\not\rightarrow}|_X$ for the maximum number of values of type X in any instantiation of $\Theta_{\not\rightarrow}$.

[3] The notion of assignment here includes implicit assignments, i.e. where a state variable does not change value because it does not appear as the target of an explicit assignment.

Theorem 5.2 *Suppose a frame* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$ *and* $X \in \Omega_{=} \cup \Omega_{\leq}$ *are such that:*

- *there is no* $a : B \rightarrow B' \in \Theta_{\rightarrow}$ *with* $X \trianglelefteq B'$*, and*
- *there is no* $a : B \rightarrow B' \in \Theta_{\rightarrow} \setminus \Theta_{\mathrm{rw}}$ *with* $X \trianglelefteq B$*.*

Then $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$*-reachability is reducible to*

$$\left\{ \left(\begin{array}{l} \Omega \setminus \{X\}, \Gamma \setminus \{=_X / \leq_X \colon X \times X \rightarrow \mathit{Bool}\}, \\ \Theta_{\rightarrow}[\mathit{Enum}_n/X], \Theta_{\mathrm{rw}}[\mathit{Enum}_n/X], I \setminus \{X\} \end{array} \right) \text{-reachability} \mid n \in \mathcal{Z}^+ \right\}$$

Proof. Consider an instance $\Theta_{\not\rightarrow}$, R, $b : \mathit{Enum}_n$, i, j, $\langle \phi_Y \mid Y \in \Omega_{\emptyset} \rangle$ of $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$-reachability.

Let $n = |\Theta_{\not\rightarrow}|_X$. Without loss of generality, we can assume $n \geq 1$. To construct an instance of reachability for the frame

$$\left(\begin{array}{l} \Omega \setminus \{X\}, \Gamma \setminus \{=_X / \leq_X \colon X \times X \rightarrow \mathit{Bool}\}, \\ \Theta_{\rightarrow}[\mathit{Enum}_n/X], \Theta_{\mathrm{rw}}[\mathit{Enum}_n/X], I \setminus \{X\} \end{array} \right)$$

we apply the ideas in [8] and [14, Chapters 5 and 6], extending them to the case $X \in \Omega_{\leq}$. In any state, values of type X in $\Theta_{\not\rightarrow}$ can be mapped to $\{1, \ldots, n\}$ isomorphically, i.e. so that $=_X$ or $\leq_X$ (respectively) corresponds with $=$ or $\leq$ on $\{1, \ldots, n\}$. Arrays whose index types contain X can then be abstracted by keeping only components whose indices do not contain any values of type X which do not occur in the current instantiation of $\Theta_{\not\rightarrow}$. □

The final result in this section is straightforward. It states that any instance of reachability which contains an array whose index type or value type is *Unit* can be reduced to an instance in which that array is removed. In the former case, the array variable reduces to a basic variable; in the latter, there is only one possible array contents.

Proposition 5.3 *Suppose* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$ *is a frame, and* $a : B \rightarrow B' \in \Theta_{\rightarrow}$ *is such that either* $B = \mathit{Unit}$ *or* $B' = \mathit{Unit}$*. Then* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$*-reachability is reducible to* $(\Omega, \Gamma, \Theta_{\rightarrow} \setminus \{a : B \rightarrow B'\}, \Theta_{\mathrm{rw}} \setminus \{a : B \rightarrow B'\}, I)$*-reachability.*

6 Decidability results

Theorem 6.1 $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$*-reachability is decidable if:*

(i) $\Omega_{\emptyset} = \{\}$,

(ii) $\Theta_{\rightarrow}$ *contains no cycles,*

(iii) *for any* $a : B \to B' \in \Theta_{\to}$, *either*
- $B \in \Omega_{=}$ *and* $B' ::= \sum_{i=1}^{n_a}(\mathit{Unit} \mid X)$, *or*
- $B \in \Omega_{\leq}$ *and* $B' ::= \sum_{i=1}^{n_a} \mathit{Unit}$,

(iv) *for any* $X \in \Omega$, *there exists a unique* $a_X : X \to \mathit{Tgt}(a_X) \in \Theta_{\to}$, *and*

(v) *for any* $X \in \Omega$, *either*
- $a_X : X \to \mathit{Tgt}(a_X) \in \Theta_{\to} \setminus \Theta_{\mathrm{rw}}$, *or*
- *there exists* X' *such that* $X \trianglelefteq \mathit{Tgt}(a_{X'})$.

Proof. Suppose $\Theta_{\not\to}$, R, $b : \mathit{Enum}_n$, f, g is an instance of $(\Omega, \Gamma, \Theta_{\to}, \Theta_{\mathrm{rw}}, I)$-reachability.

For any $X \in \Omega_{=}$, we define

$$k_X = |\Theta_{\not\to}|_X \qquad m_X = 1$$

For any $X \in \Omega_{\leq}$, we define

$$k_X = |\Theta_{\not\to}|_X \qquad m_X = |\Theta_{\not\to}|_X + 1$$

We shall use the following operators on quasi orders: finitary products; powerset with cardinality at most 1, where the empty set is incomparable with singletons;[4] the set of all finite bags with the pointwise subbag ordering; the set of all finite sequences with the pointwise noncontiguous subsequence ordering. Each of these operators preserves the wellness property.

For any $X \in \Omega$, we define a well-quasi-order as follows. The sets $\{j \mid \mathit{Tgt}(a_X)_j = \mathit{Unit}\}$ have the identity orderings. We use the operator *Bag* or *Seq* depending on whether $X \in \Omega_{=}$ or $X \in \Omega_{\leq}$ (respectively).

$$\mathcal{R}_X = \prod_{i \in \{1,\ldots,k_X\}} \mathcal{P}_{\leq 1}(\{j \mid \mathit{Tgt}(a_X)_j = \mathit{Unit}\} \times \mathcal{S}_X) \times \prod_{i \in \{1,\ldots,m_X\}} \mathit{Bag}/\mathit{Seq}(\{j \mid \mathit{Tgt}(a_X)_j = \mathit{Unit}\} \times \mathcal{S}_X)$$

The well-quasi-orders $\mathcal{S}_Y$ for $Y \in \Omega$ are defined by:

$$\mathcal{S}_Y = \left(\prod_{\mathit{Tgt}(a_X)_i = Y, j \in \{1,\ldots,k_X\}} \mathcal{P}_{\leq 1}(\mathcal{S}_X) \right) \times \left(\prod_{\mathit{Tgt}(a_X)_i = Y, j \in \{1,\ldots,m_X\}} \mathit{Bag}(\mathcal{S}_X) \right)$$

Now, let $(S, \to)$ be the transition system of $(\Omega, \Gamma, \Theta_{\to}\Theta_{\not\to}, R, I)$. We can define a mapping from S to a well-quasi-order

$$\{1, \ldots, N\} \times \prod_{X \in \Omega} \mathcal{R}_X$$

[4] This is equivalent to disjoint union (i.e. sum) with a singleton quasi order.

where $\{1, \ldots, N\}$ has the identity ordering. Let $\sqsubseteq$ be the induced well-quasi-ordering on S.

We have that $(S, \rightarrow)$ is a well-structured transition system with respect to $\sqsubseteq$. It follows that we have a reduction to the covering problem for well-structured transition systems with effective *pred*-basis and decidable ordering, which was shown decidable in [6]. □

Given an instance of reachability for a frame which does not satisfy the conditions in Theorem 4.1, the following result states that it is reducible to an instance of reachability for a frame to which Theorem 6.1 applies.

Proposition 6.2 *Suppose* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$ *is a frame which does not satisfy any of (a)–(d) in Theorem 4.1. Then* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$*-reachability is reducible to*

$$\{(\Omega^h, \Gamma^h, \Theta^h_{\rightarrow}, \Theta^h_{\mathrm{rw}}, I^h)\text{-reachability} \mid h \in H\}$$

where H *is countable, and each of the frames* $(\Omega^h, \Gamma^h, \Theta^h_{\rightarrow}, \Theta^h_{\mathrm{rw}}, I^h)$ *satisfies the assumptions of Theorem 6.1.*

Proof. By repeated application of Theorem 5.1, Theorem 5.2 and Proposition 5.3. □

Corollary 6.3 *Suppose* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$ *is a frame which does not satisfy any of (a)–(d) in Theorem 4.1. Then* $(\Omega, \Gamma, \Theta_{\rightarrow}, \Theta_{\mathrm{rw}}, I)$*-reachability is decidable.*

7 Conclusions and future work

We have obtained a complete classification into decidable or undecidable of reachability decision problems for polymorphic systems with arrays in which the only operations available on type variables are equality and linear-order predicates. The principal tools used are simulations of 2-counter machines, data-flow analysis, abstract interpretation, and the theory of well-structured transition systems.

For frames whose reachability decision problem is decidable, it would be interesting to obtain bounds on its complexity. It seems that this complexity is higher for frames which contain type variables with linear-order predicates. This corresponds to complexity of reachability for lossy channel systems being higher than that of covering for Petri nets [16].

Acknowledgements

I am grateful to Tom Newcomb, Joël Ouaknine and Bill Roscoe for helpful discussions.

References

[1] M. Bozzano and G. Delzanno, *Beyond Parameterized Verification*, Proceedings of TACAS '02, Lecture Notes in Computer Science 2280, 221–235, 2002.

[2] M. Bozzano and G. Delzanno, *Automatic Verification of Invalidation-based Protocols*, Proceedings of CAV '02, July 2002.

[3] G. Delzanno, *An Assertional Language for Systems Parametric in Several Dimensions*, Proceedings of the Workshop on Verification of Parameterized Systems (VEPAS 2001), Electronic Notes in Theoretical Computer Science 50, Elsevier, 2001.

[4] G. Delzanno, *On the Automated Verification of Parameterized Concurrent Systems with Unbounded Local Data*, Technical Report, Dipartimento Informatica e Scienze dell'Informazione, Università di Genova, 2002. Revised and extended version of [3], [1], [2].

[5] E.A. Emerson and K.S. Namjoshi, *On model checking for non-deterministic infinite-state systems*, Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS), 1998.

[6] A. Finkel and Ph. Schnoebelen, *Well-structured transition systems everywhere!*, Theoretical Computer Science 256 (1–2): 63–92, 2001.

[7] H. Garcia-Molina, *Elections in a distributed computing system*, IEEE Transactions on Computers 31 (1): 48–59, 1982.

[8] R.S. Lazić, T.C. Newcomb and A.W. Roscoe, *On model checking data-independent systems with arrays without reset*, Theory and Practice of Logic Programming 4 (5 & 6): 659–693, Cambridge University Press, 2004.

[9] R.S. Lazić, T.C. Newcomb and A.W. Roscoe, *On model checking data-independent systems with arrays with whole-array operations*, Proceedings of 25 Years of CSP, July 2004, to appear.

[10] R.S. Lazić, T.C. Newcomb and A.W. Roscoe, *Polymorphic systems with arrays, 2-counter machines and multiset rewriting*, Proceedings of the International Workshop on Verification of Infinite-State Systems (INFINITY 2004), Electronic Notes in Theoretical Computer Science.

[11] R. Lazić and A.W. Roscoe, *Verifying determinism of concurrent systems which use unbounded arrays*, Proceedings of the 3rd International Workshop on Verification of Infinite-State Systems (INFINITY '98), Report TUM-I9825, 2–8, Technical University of Munich, July 1998.

[12] I. A. Lomazova, *Nested Petri Nets: Multi-level and Recursive Systems*, Fundamenta Informaticae 47, 283–294, IOS Press, 2001.

[13] J.C. Mitchell, *Type Systems for Programming Languages*, in [17], 365–458.

[14] T.C. Newcomb, *Model Checking Data-Independent Systems with Arrays*, D.Phil. thesis, Computing Laboratory, Oxford University, 2003.

[15] J.-F. Raskin and L. Van Begin, *Petri Nets with Non-blocking Arcs are Difficult to Analyse*, Proceedings of the International Workshop on Verification of Infinite-State Systems (INFINITY 2003), Electronic Notes in Theoretical Computer Science.

[16] Ph. Schnoebelen, *Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity*, Information Processing Letters 83 (5): 251–261, 2002.

[17] J. van Leeuwen (editor), *Formal Models and Semantics*, Handbook of Theoretical Computer Science, volume B, Elsevier, 1990.